Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.5, Oct. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.5.1151

메모리 주소 변환 공격을 탐지하기 위한 Snoop기반의 커널 검사 시스템*

김 동 욱,†‡ 김 지 훈, 박 진 범, 김 진 목
삼성전자 소프트웨어센터

A Snoop-Based Kernel Introspection System against Address Translation Redirection Attack*

Donguk Kim,†‡ Jihoon Kim, Jinbum Park, Jinmok Kim
Software R&D Center, Samsung Electronics

요 약

TrustZone의 시큐어 타이머를 이용한 커널 루트킷 탐지 시스템은 커널로부터 분리되고, 독립된 환경에서 커널을 보호할 수 있기 때문에 모니터링 시스템의 무결성을 보장할 수 있다. 하지만, 물리 메모리 주소를 기반으로, 커널 메모리를 주기적으로 모니터링하기 때문에 일시적인 공격에 취약하며, 페이지 테이블을 변조하여 가상-물리 메모리 주소 변환을 조작하는 공격을 탐지할 수 없다는 단점이 있다. 이를 해결하기 위해, 본 논문에서는 Snoop기반의 커널 검사 시스템을 제안한다. 이 시스템은 커널 메모리를 실시간으로 보호하기 위해 Snooper를 이용하여 모니터링 하며, 프로세스의 컨텍스트 스위칭 시마다 커널 페이지를 검사하여 주소 변환 공격 여부를 검증한다. 커널 검사 시스템은 TizenTV에서 구현되었으며, 실험결과들은 제안된 커널 검사 시스템이 커널 메모리 및 해당 페이지 테이블을 실시간으로 보호하며, 4.67%정도의 성능만 저하시킨다는 것을 보여준다.

ABSTRACT

A TrustZone-based rootkit detecting solution using a secure timer ensures the integrity of monitoring system, because ARM TrustZone technology provides isolated environments from a monitored OS against intercepting and modifying invoke commands. However, it is vulnerable to transient attack due to periodic monitoring. Also, Address Translation Redirection Attack (ATRA) cannot be detected, because the monitoring is operated by using the physical address of memory. To ameliorate this problem, we propose a snoop-based kernel introspection system. The proposed system can monitor a kernel memory in real-time by using a snooper, and detect memory-bound ATRA by introspecting kernel pages every context switch of processes. Experimental results show that the proposed system successfully protects the kernel memory without incurring any significant performance penalty in run-time.

Keywords: Platform security, System security, TrustZone, Page Table, ATRA

I. 서 론

커널 루트킷은 운영체제를 무력화시키기 위해, 커널 메모리에 악성 코드를 삽입할 수 있다. 이러한 커널 루트킷은 커널권한을 탈취하여 루트킷 탐지 모니터링 시스템까지 무력화 시킬 수 있어서, 사이버

Received(07. 25. 2016), Modified(08. 19. 2016), Accepted(08. 29. 2016)
* 본 연구는 삼성전자 소프트웨어센터의 지원 및 관리로 수행하였습니다.
† 주저자, donguk14.kim@samsung.com
‡ 교신저자, donguk14.kim@samsung.com(Corresponding author)

공격에 사용되어 왔다. 따라서 커널과 동일한 공간으로부터 분리되고, 독립된 환경에서 커널의 무결성을 보장해주는 커널보호 시스템의 필요성이 대두되고 있다.

ARM TrustZone[1]은 2개의 가상 CPU 코어 및 모니터를 통해 normal world와 secure world 사이의 안전한 컨텍스트 스위칭 방법을 제공한다. 이를 이용하여 독립된 공간에서 안전한 실행환경을 구축할 수 있고, secure world 메모리에 대한 접근도 차단할 수 있다. 커널보호 연구 중 하나인 TrustZone 기반 연구는 ARM TrustZone의 특징을 이용하여, 커널과 동일한 메모리 주소 공간으로부터 독립된 공간에서 커널 무결성을 검사하는 방법이다. 하지만, 이미 제안된 방법인 TZ-RKP[2]와 SPROBES[3]는 normal world내의 프로세스로부터 검사 요청을 받아서 secure world내에서 커널 메모리를 검사하기 때문에 커널검사 시스템을 완전히 독립시킬 수 없다. SensePost[4]에 공개된 것과 같이, 루트킷이 man-in-the middle 공격을 통해 검사 요청 메시지를 변조시킨다면, 커널 메모리에 대한 검사를 수행할 수 없게 된다. 이러한 문제를 해결하기 위해, EKI[5]는 TrustZone의 시큐어 타이머를 이용한 커널 검사 시스템을 제안하였다. 하지만, 커널 메모리를 주기적으로 모니터링하기 때문에, 만약 루트킷이 커널 메모리에 대한 공격 후, 커널검사 시스템의 모니터링 주기 안에 메모리를 정상상태로 돌려놓는다면 이를 탐지할 수 없다. 이와 같이, 짧은 시간 안에 공격 후, 그 공격 흔적을 지우는 방법을 일시적인 공격(transient attack)이라 하며, 이는 실시간 모니터링으로 탐지가 가능하다. EKI에서 실시간으로 모니터링 하는 효과를 내기 위해 주기를 짧게 설정하고 모니터링 블록의 크기를 크게 설정하는 방법도 제시하였으나, 단말의 성능이 저하되는 문제가 있었다. 또한 물리 메모리 주소만을 사용하여 검사하기 때문에 ATRA[6]에서 소개된 메모리에 대한 페이지 테이블을 변조하여 가상-물리 메모리 주소 변환을 조작하는 공격을 탐지할 수 없다는 단점이 있다. 만약 루트킷이 커널 메모리에 대한 페이지를 변조하여 커널의 가상 주소를 악성코드가 삽입된 물리 주소로 매칭 시키면, 커널동작 시, 악성코드가 실행될 수 있다.

이러한 이유로, 본 논문에서는 실시간으로 커널 메모리를 검사하고, ATRA를 탐지할 수 있는 Snoop기반의 커널 검사 시스템(SKIS, A Snoop-based Kernel Introspection System against Address Translation Redirection Attack)을 제안한다. SKIS는 시스템 버스의 트래픽을 지속적으로 모니터링 할 수 있는 Bus Snooper[7]를 이용하여 실시간으로 커널 메모리의 변조를 감시하고, 일시적인 공격도 탐지할 수 있다. SKIS에서는 Snooper에 대한 설정이 TrustZone 내에서만 가능하도록 설계하였기 때문에, 커널 메모리에 대한 보호를 독립적이고 실시간으로 수행할 수 있다. 또한, SKIS는 프로세스의 컨텍스트 스위칭 시마다 커널 페이지를 검사하여 주소 변환 공격 여부를 검증한다. 모든 프로세스는 Page Global Directory(PGD)라고 불리는 루트 페이지 테이블의 커널 페이지에 Master PGD의 커널 코드 페이지를 복사한다. 따라서 모든 프로세스의 커널 코드 영역에 대한 페이지는 동일하며, 컨텍스트 스위칭 시마다 이를 검사함으로써, ATRA를 탐지할 수 있다.

본 논문에서 제안하는 커널 검사 시스템은 프로세스의 컨텍스트 스위칭 시마다 동작하기 때문에 보안성과 함께 시스템의 성능 저하 수준이 고려되어야 한다. 이에 대한 평가를 위해 실제 사용되는 단말인 Samsung TizenTV에서 SKIS를 구현하여 검증하였으며, 커널 메모리 변조를 정확하게 탐지한다는 것과 성능 저하가 크지 않다는 것을 확인하였다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 논문과 관련된 배경 지식들에 대해 설명하며, 3장에서는 ATRA에 대한 설명과 함께 논문에서 막고자 하는 공격모델을 제시한다. 4장에서는 공격모델에 대한 탐지 시스템을 제시하고, 5장에서는 제안된 시스템에 대한 평가결과를 보여준다. 마지막으로 5장에서는 결론 및 향후 연구방향을 제시한다.

II. 배경지식

2.1 ARM TrustZone

Fig. 1.에서 보는 바와 같이, TrustZone은 2개의 world 사이의 안전한 컨텍스트 스위칭 방법을 제공한다. 사전에 각 world에서 동작될 수 있는 하드웨어 자원들이 구분되기 때문에 다른 world의 프로세서, 버스, 메모리, 주변장치 등에 대한 직접적인 접근이 차단된다. 따라서 TrustZone 이라는 안전한 실행환경을 제공할 수 있으며, 이를 이용하기 위해서는 normal world에서 secure world로 모드를 전환해야 한다. 모드 전환의 방법은 IRQ(Interrupt ReQuest)나 FIQ(Fast

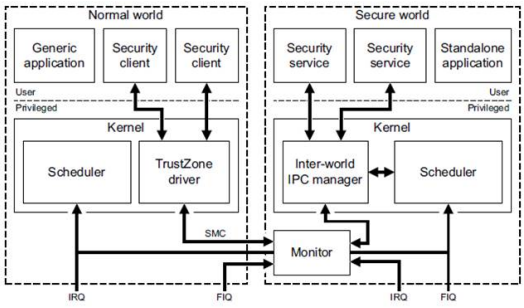

Fig. 1. ARM TrustZone software architecture[1]

Interrupt reQuest)를 이용하여 모니터 모드로 변경하는 방법 또는 TrustZone 드라이버를 통해 SMC 인스트럭션을 이용하는 방법이 있다. Security client(client application)가 secure world내의 security service(trusted application)를 이용하기 위해서는 SMC 인스트럭션 방식을 이용하여야 한다.

2.2 TrustZone 기반 커널 보호 연구 - EKI

EKI는 완전히 독립된 공간에서 커널 검사 시스템을 운영할 수 없다는 TrustZone 기반 연구의 단점을 극복하기 위해 TrustZone의 시큐어 타이머를 이용한 주기적인 커널 메모리 검사 시스템을 제안하였다. 빌드타임 시, 커널 원본 이미지인 vmlinux 파일을 이용하여 보호영역의 메모리를 나타내는 레퍼런스 값을 측정하였고, 런타임 시에는 secure world내에 각 보호영역에 대한 페이지를 생성하여 보호영역의 런타임 메모리 값을 측정하였다. Secure world내의 trusted application이 normal world상의 프로세스 호출 없이 독립적으로 동작하기 위해서 TrustZone의 시큐어 타이머를 구현하여 주기적으로 보호영역에 대한 검사를 수행하였다. 따라서 커널 루트킷의 공격으로부터 안전한 독립된 환경에서 동작될 수 있다. 그러나 주기적인 검사 방법으로 인해 일시적인 공격에 취약하다.

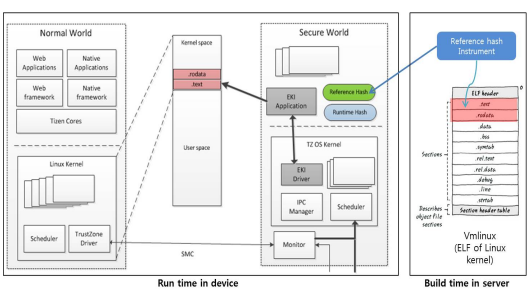

Fig. 2. EKI architecture[5]

2.3 Snooper

Snooper는 CPU, I/O 장치, 기억장치들을 상호 연결해주는 시스템 버스의 트래픽을 지속적으로 모니터링 할 수 있는 하드웨어이다. 시스템 버스의 구성하는 버스 중 하나인 Control 버스의 제어 신호가 필터링 규칙에 해당되는 것이면 Data 또는 Address 버스의 데이터를 기록하거나 차단할 수 있다. Snooper의 설계에 따라 트래픽에 대한 필터링 규칙을 하드웨어 자체에 미리 설정하거나 드라이버를 통해 설정이 가능하며, 모니터링 결과에 대한 리포팅 방식의 설정도 가능하다.

Fig. 3.은 Snooper를 이용한 커널 모니터링 시스템 중 하나인 Vigilare[8] 시스템이다. SnoopMon 방식을 사용하는 Vigilare의 Snooper는 커널의 read-only 영역에 대한 write traffic이 탐지되면 이를 차단하며, SnapMon 방식을 사용하는 Vigilare의 Snooper는 read-only 영역에 대한 write traffic을 필터링한다. 필터링된 traffic을 verifier 프로세서에게 전달하면, verifier에서는 전달된 traffic 정보를 바탕으로 해당 영역에 대한 호스트 시스템의 메모리 값을 읽어 들인 후, 미리 저장해 놓은 원본 값과 비교하여 변조여부를 판단한다. 이와 같이, 호스트 시스템과 독립적으로 동작할 수 있는 하드웨어 Snooper의 특성을 이용하여 커널 보호 시스템의 보안성을 향상시킬 수 있다.

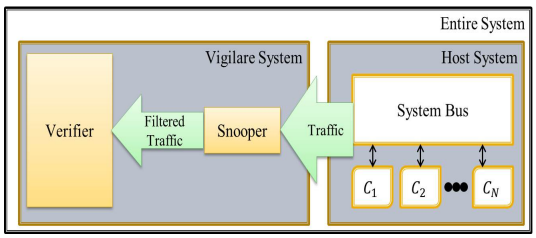

Fig. 3. Vigilare system[8]

III. 공격 모델

커널 루트킷은 커널 권한을 가진 채로 공격을 수행할 수 있기 때문에 운영체제의 모든 프로세스를 공

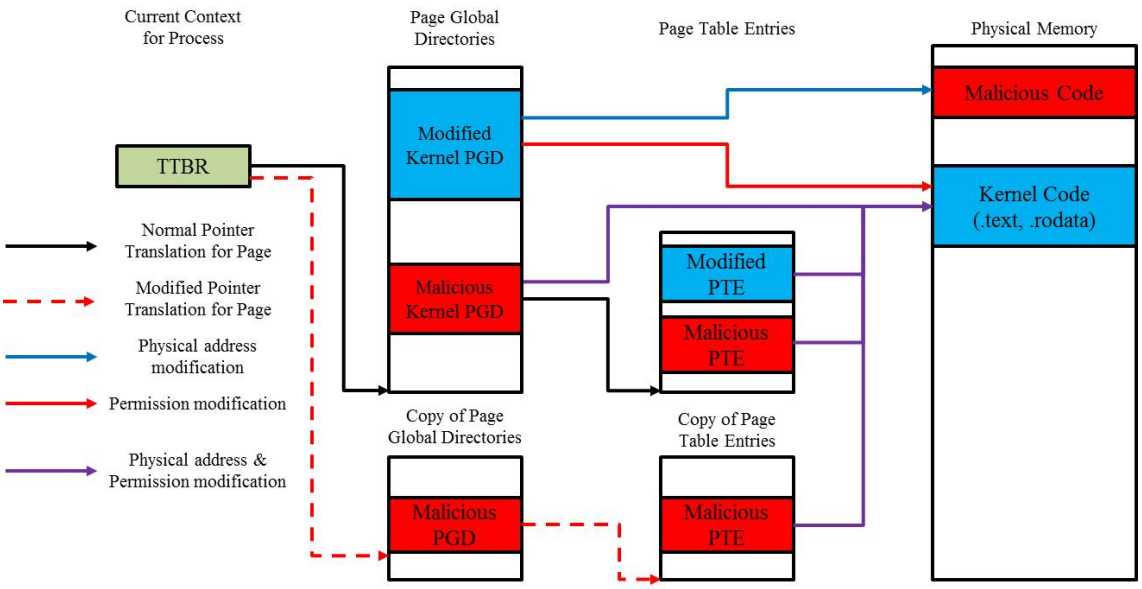

Fig. 4. Attack types of ATRA

격할 수 있다. 본 연구에서는 32bit 리눅스 운영체제를 사용하는 단말에 커널 루트킷이 이미 설치되어 있고, Poker[9]에서 프로파일링된 결과에 따라 루트킷의 기본적인 공격특성인 코드 인젝션 및 시스템 콜 후킹을 통해 ARM TrustZone에 의해 보호되는 secure world를 제외한, normal world의 커널 코드 영역을 변조할 수 있다고 가정한다. 또한, EKI에서 탐지할 수 없었던 일시적인 공격을 추가하여, 메모리 변조 후에 정상상태로 복구할 수 있다고 가정한다. 따라서 커널 메모리에 대한 변조를 실시간으로 검사하지 않는다면 이를 탐지할 수 없다.

하지만, EKI와 같은 기존의 커널 검사 시스템은 물리주소를 기반으로 동작하기 때문에, 가상 주소에서 물리 주소로의 변환 정보를 담고 있는 페이지 테이블을 변조하는 경우 탐지 시스템을 우회할 수 있다. 따라서 본 논문에서 제시한 커널 루트킷은 커널 메모리에 대한 직접적인 변조 이외에 주소변환 공격인 ATRA를 이용하여, 커널 메모리에 대한 페이지를 변조할 수 있다고 가정한다. Fig. 4.에서 보는 바와 같이, 세 가지 방식의 커널 페이지 변조방식이 있으며, 이를 통해 커널 동작 시, 악성코드를 실행시킬 수 있다.

첫째로, ATRA는 커널 PGD의 커널 코드 주소를 악성 코드가 로딩 되어 있는 메모리 주소로 변조할 수 있다. 이 경우, 커널 동작 시에 정상적인 커널 코드 대신, 악성코드가 실행되며, 기존 시스템에서는 커널 코드에 대한 메모리만 검사하기 때문에 이에 대한 우회가 가능하다.

둘째로, ATRA는 커널 코드 메모리를 수정할 수 있는 커널 PGD의 권한을 변조할 수 있다. 이 경우, 커널 루트킷은 커널 코드가 로딩 되어 있는 메모리에 접근 가능하며, 이에 대한 수정도 가능하다. 두 번째 방식의 경우, 커널 코드에 대한 Page Table Entry(PTE)는 존재하지 않기 때문에 PTE를 변조하는 경우는 고려하지 않는다.

셋째로, ATRA는 악성 커널 PGD 및 PTE를 생성하여 커널 코드 메모리를 변조할 수 있다. 앞서 말한 가상-물리 주소 변환 테이블 및 권한을 변조하여 이와 같이 공격할 수 있으며, 두 번째 방식과 동일한 공격결과를 유도한다. 컨텍스트 스위칭 시, 악성 커널 PGD를 실행시키기 위해서는 Translation Table Base Register(TTBR)에 저장되어 있는 PGD 주소의 포인터를 변조해야 한다. 커널 루트킷은 TTBR의 저장된 값을 조작하여 컨텍스트 스위칭 시마다 악의적으로 생성된 악성 커널 PGD를 참조하도록 공격할 수 있는데, 이를 register-bound ATRA[6]라고 한다. 본 연구에서는 register bound ATRA는 메모리 주소 변환 공격에 대한 스케줄러 기반의 방법[10]에 의해 탐지된다고 가정한다.

IV. 탐지 방법 설계

본 연구에서는 앞서 얘기한 일시적인 공격에 대한 EKI의 단점을 보완하고 물리주소를 기반으로 동작하는 커널 검사 시스템을 우회 가능한 ATRA에 대한 탐지를 목표로 Snoop기반의 커널 검사 시스템을 설계하였다.

4.1 보호 영역 주소 설정

Fig. 5.는 SKIS의 전반적인 아키텍처를 나타낸다. Secure world내에 저장되어 있는 레퍼런스는 보호 영역에 대한 물리 메모리 주소를 나타내는 값이다. 만약, 커널이 루트킷에 의해 변조된 상태라면, 런타임 시에 레퍼런스를 측정하는 것은 무결성이 보장되지 않기 때문에 빌드타임 시, 레퍼런스 측정도구를 통해 보호 영역에 대한 주소 정보를 추출하고 이를 저장한다. 부팅 시, secure world내에 저장되어 있는 레퍼런스를 참조하여, 미리 단말에 설치되어 있는 메모리 프로텍터 및 페이지 테이블 프로텍터의 보호영역이 설정되며, 메모리 프로텍터 초기화 시에, secure world내의 커널 모듈로 Snooper 드라이버가 삽입되어 Snooper에 대한 보호영역 및 동작방식을 설정한다. 런타임 시에는 각 프로텍터가 설정된 보호영역에 대한 모니터링을 수행한다.

SKIS에서는 빌드타임 시, 보호영역에 대한 주소정보를 추출하기 위해 EKI와 마찬가지로 vmlinux 파일을 활용하였다. ELF 파일과 동일한 포맷으로 구성되어 있는 vmlinux 파일은 데이터의 특성에 맞게 여러 개의 섹션 및 심볼들로 구성되어 있다. 각 섹션 별로 이름, 가상시작주소, 오프셋, 크기, 접근권한 정보가 포함되어 있으며, 각 심볼 별로는 이름, 가상주소, 타입, 크기 등에 대한 정보도 포함되어 있기 때문에 커널 코드들이 저장되어 있는 text 영역, read-only initialized 데이터가 존재하는 rodata 영역의 정보 및 Master swapper_pg_dir 심볼을 통해 Master PGD영역에 대한 정보를 구할 수 있다.

리눅스는 가상 메모리를 사용하는 시스템이며, 주소영역이 페이지 단위로 관리되기 때문에 가상 메모리 주소가 연속적으로 이어져 있어도 물리 메모리 주소는 분산되어 있을 수 있다. 하지만, SKIS에서 보호하는 영역은 ZONE_NORMAL 이라고 명칭 된 영역 안에서 가상메모리와 물리메모리가 일대일로 매칭 되기 때문에 연속적으로 로딩 된다. 리눅스 커널의 물리메모리 시작 주소인 PHYS_OFFSET은 커널 빌드 옵션 설정 파일인 .config 파일에 존재하고, 리눅스 커널의 가상메모리 시작 주소인 PAGE_OFFSET 정보는 하드웨어 정보가 포함된 dtb.bin 파일에 존재한다. 따라서 레퍼런스 측정도구에서 위 파일들과 vmlinux 파일을 이용하여 각 정보를 추출하고 아래 식에 대입하면, 보호영역의 가

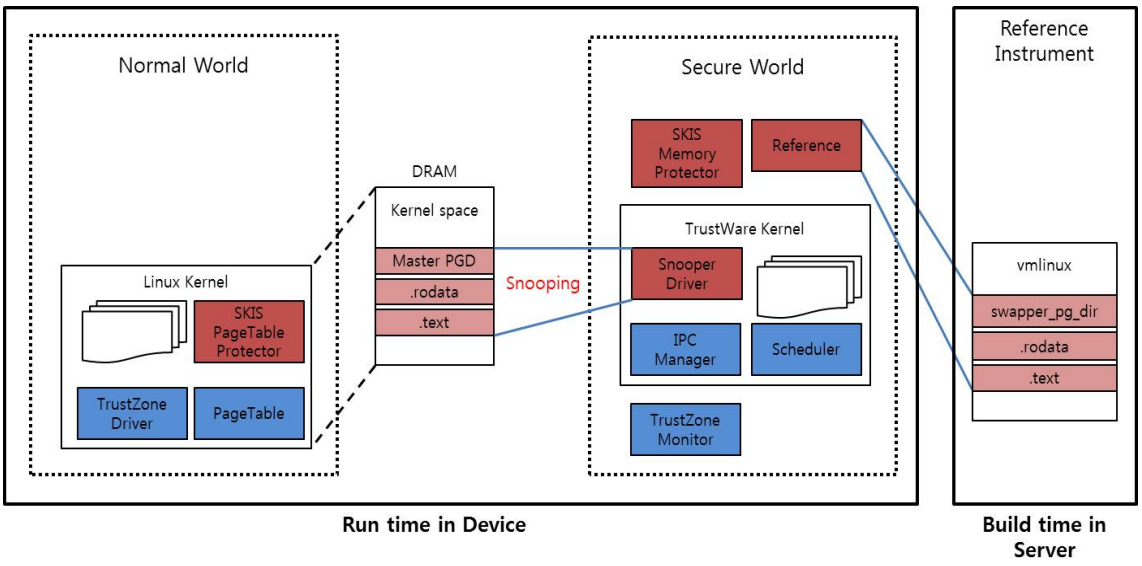

Fig. 5. SKIS architecture

상메모리 주소만으로도 물리메모리 주소를 구할 수 있다.

$$\text{물리주소} = \text{가상주소} - \text{PAGE\_OFFSET} + \text{PHYS\_OFFSET} \quad (1)$$

빌드타임 시에 측정된 보호영역의 물리 메모리 주소는 메모리 및 페이지 테이블 프로텍터의 보호영역으로 설정되어 런타임 시, 무결성 검증 수행에 이용된다.

4.2 메모리 보호

SKIS의 두 가지 주요 컴포넌트 중 하나는, 커널 코드가 로딩 되어 있는 메모리를 모니터링 하는 메모리 프로텍터이다. 메모리 프로텍터는 TrustZone에서 동작하며, 빌드타임 시에 측정된 레퍼런스를 바탕으로 부팅 시, Snooper 드라이버를 통해 Snooper의 보호영역을 설정한다. 설정된 Snooper는 CPU와 DRAM 사이의 트랜젝션을 통해 보호영역의 변조를 차단하고 이를 메모리 프로텍터로 전달하는 기능을 수행한다.

Fig. 6.에서 보는 바와 같이, SKIS에서의 Snooper는 Special Function Register(SFR)들과 Write Traffic Detector(WTD)로 구성된다. SFR은 Snooper의 동작방식과 보호영역의 설정 및 탐지결과의 저장용도로 쓰이며, Advanced Microcontroller Bus Architecture(AMBA)[11]에 정의되어 있는 Advanced Peripheral Bus(APB)를 통해 설정이 가능하다. WTD는 Advanced eXtensible Interface(AXI)를 통해 들어오는 트래픽의 write영역 주소를 필터링 한다. SFR에 설정된 보호영역이라면 이를 차단하고, 탐지결과 저장 후, FIQ를 통해 시스템에 등록된 SKIS 드라이버 함수를 호출한다. SKIS 드라이버는 SFR에 저장된 탐지결과를 시스템 이벤트에 포함시켜서 secure world 운영체제의 이벤트 처리 루틴을 통해 메모리 프로텍터로 전달한다. 이렇게 구현된 메모리 보호기능은 독립된 환경에서 Snooper를 이용하여 동작하기 때문에 시스템의 성능저하 없이 실시간으로 메모리를 보호할 수 있다.

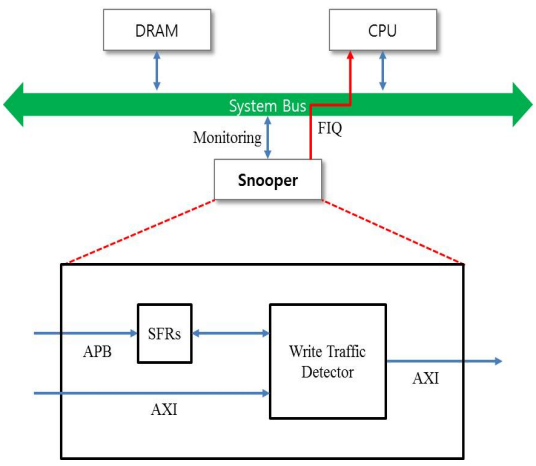

Fig. 6. Snooper architecture of SKIS

4.3 페이지 테이블 보호

두 번째 주요 컴포넌트는 커널 코드 메모리 영역에 대한 페이지를 보호하는 페이지 테이블 프로텍터이다. 페이지 테이블 프로텍터는 커널 코드의 일부로 동작하며, 프로세스의 컨텍스트 스위칭 시마다 커널 페이지를 검사한다. 1장에서 언급했듯이, 모든 프로세스는 Master PGD의 커널 코드 페이지를 자신의 PGD에 복사하여 생성되기 때문에 Master PGD가 변조되지 않는 한, 모든 PGD의 커널 코드 영역은 동일하다. 따라서 컨텍스트 스위칭 시마다 이 특징을 이용하여 ATRA를 탐지할 수 있다.

Fig. 7.은 컨텍스트 스위칭 시마다 PGD 검사를 수행하기 위한 페이지 테이블 프로텍터의 후킹 과정을 나타낸 것이다. 부팅 시, 컨텍스트 스위칭 함수인 'check_and_switch_context'의 시작 인스트럭션을 복사하여 후킹 함수의 마지막 부분에 삽입한다. 삽입된 부분 뒤에는 'check_and_switch_context'의 시작 인스트럭션 이후 부분이 정상적으로 실행되도록 점프 인스트럭션을 삽입한다. 마지막으로 후킹 함수를 호출하는 인스트럭션으로 시작 인스트럭션을

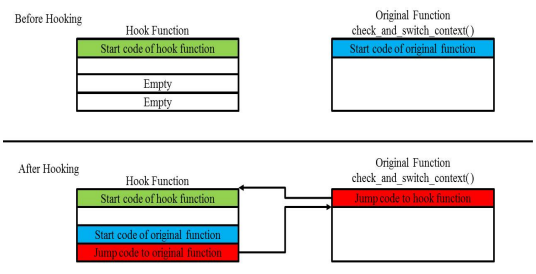

Fig. 7. Hooking process by page table protector

대체하면, 후킹이 완료된다. 이 때 사용되는 인스트럭션의 사이즈는 'check_and_switch_context'의 시작 인스트럭션 사이즈와 같다.

앞서 설명한 것과 같이, 후킹이 완료된 이후에는 'check_and_switch_context' 함수가 호출될 때마다 후킹 함수가 먼저 호출된다. 이 때, 레지스터 r0부터 r3까지 'check_and_switch_context' 함수의 arguments가 저장되는데 이는 강제적인 후킹에 의해 훼손될 가능성이 있으므로 메모리에 저장한다. 이후에, 프로세스의 task_struct내에 포함되어 있는 mm_struct 정보를 이용하여 PGD의 커널 코드 영역에 접근하여 Master PGD와의 비교 후, 현재 프로세스의 실행여부를 결정한다. Master PGD는 'swapper_pg_dir' 심볼을 통해 주소를 알 수 있으며, 정적 영역이기 때문에 Snooper를 통해 보호가 가능하다. 실행 여부가 결정된 이후에는 메모리에 저장하였었던 r0에서 r3까지의 레지스터 값을 복구하고, 'check_and_switch_context' 함수의 시작 인스트럭션을 수행하면 페이지 프로텍터의 동작이 완료된다. 컨텍스트 스위칭 시마다 이와 같은 동작과정을 통해 ATRA의 탐지가 가능하다.

V. 평 가

SKIS의 보안성 및 성능을 평가하기 위해서 3장에서 제시한 threat model의 공격 탐지여부와 운영체제의 성능저하 측면에서 이를 평가하였다. 모든 실험은 리눅스 3.10.30 커널과 퀴드코어 ARMv7 CPU가 내장되어 있는 실제 단말인, Samsung Tizen TV상에서 수행되었다.

5.1 보안성 평가

본 실험에서는 Samsung Tizen TV용 threat model을 직접 구현하여 커널 메모리를 변조하였다. 일시적인 공격을 구현하기 위해 text와 rodata 섹션의 메모리 값을 변조한 이후, 다시 원본 메모리 값으로 복구하도록 설계하였다. ATRA공격의 경우, 불특정 프로세스들의 PGD의 커널 코드 영역 페이지 값을 다른 주소로 변조하거나 권한을 변경하도록 설계하였다. 평가 방법은 공격 주소와 탐지 주소가 같은지 확인하여 해당 공격에 대한 탐지 성공여부를 판단하였고, 일시적인 공격에 대한 실시간 탐지 여부를 판단하기 위해 공격 시도에서부터 이를 탐지하기

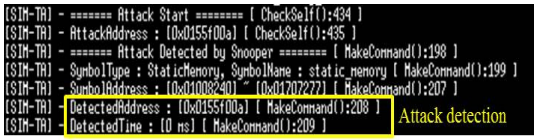

Fig. 8. Transient attack detection by SKIS

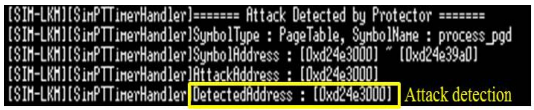

Fig. 9. ATRA detection by SKIS

까지 걸리는 시간을 탐지시간이라고 정의하여, 0ms이라면 이를 실시간 탐지라고 판단하였다.

Fig. 8.은 일시적인 공격에 대한 탐지결과를 나타낸다. 탐지 주소가 공격 주소인 0x0155f00a와 동일한 결과를 통해, 해당 공격이 탐지되었음을 확인할 수 있고, 탐지시간이 0ms인 결과를 통해, SKIS의 메모리 프로텍터에 의해 실시간으로 탐지된다는 것을 확인할 수 있었다. Fig. 9.는 ATRA에 대한 탐지결과를 나타낸다. 프로세스 PGD 내의 커널 코드 영역 주소인 0xd24e3000 ~ 0xd24e39a0 주소 영역 안에 공격 주소인 0xd24e3000이 포함되는 것으로 PGD에 대한 공격이 이루어졌음을 확인할 수 있고, 이는 탐지 주소와 동일하기 때문에 SKIS의 페이지 테이블 프로텍터에 의해 탐지된다는 것을 확인할 수 있었다.

5.2 성능 평가

SKIS의 페이지 테이블 프로텍터는 프로세스의 컨텍스트 스위칭 시마다 동작하기 때문에 운영체제의 성능 저하를 발생시킬 수 있다. 운영체제의 성능은 벤치마크에서 측정된 score값을 통해 표현할 수 있으며, SKIS가 동작하기 전과 후의 벤치마크 score 값을 비교하면, 동작 시의 성능이 정상적인 성능 대비 얼마인지 측정할 수 있고, 성능 저하 비율도 계산할 수 있다.

그래픽과 네트워크 접근 등과 같은 일반적인 시스템 이벤트들이 발생하는 환경에서, SKIS의 동작으로 인한 성능 저하를 측정하기 위해 ARM에서 동작 가능한 Streamline[12], CoreMark[13], Unixbench[14], Linpack[15], nbench[16]등 5개의 벤치마크를 사용하여 score값을 측정하였다.

Table 1. Measurement Information of BenchMark

BenchMark	Measurement Information
Streamline	CPU Activity, Branch, Bus, Cache, Clock
Coremark	Read/Write Operations, Integer Operations, Control Operations
UnixBench	String Handling, Floating Point Operations, Execl Throughput, File Copy, Pipe Throughput, Context Switching, Process Creation, Shell Scripts, System Call Overhead, Graphical Tests
Linpack	Floating Point Operations
nbench	Numeric Sort, Floating Point Operations, String Sort, Write Operation

Table 1.에서 보는 바와 같이, 각 벤치마크들은 다양한 지표를 통해 운영체제의 성능에 대한 score값을 측정하게 되고, 본 실험에서는 여러 성능 지표를 고려하기 위해 다양한 벤치마크들을 사용하였다.

Fig. 10.은 벤치마크에서 측정한 score값을 통해 SKIS 동작 시의 성능이 운영체제의 정상적인 동작 시의 성능과 비교하여 비율이 얼마인지 나타낸 그래프이다. 이 측정 결과를 통해, Streamline으로 측정하였을 경우, 다른 벤치마크들의 측정결과보다 큰 성능저하가 나타나지만 측정값이 크지 않음을 확인할 수 있었다. 이는 Table 2.에서 보는 바와 같이, 다른 연구들의 성능저하와 비교하였을 때 SKIS가 성능저하 측면에서 우수하다는 것을 보여준다.

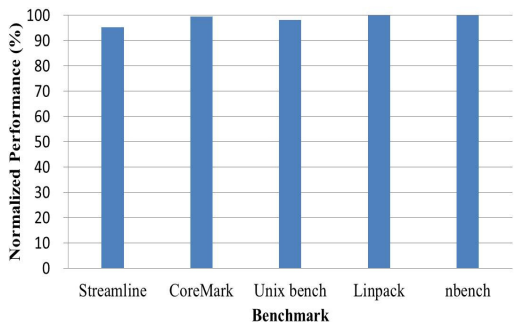

Fig. 10. Performance that measured by benchmarks

Table 2. Maximum Overhead per Monitor

Monitor	Defense Type	Protection Area	Maximum Overhead
TZ-RKP [2]	Event driven	Static and Dynamic memory	7.65%
EKI [5]	Active monitoring	Static memory	6.09%
SKIS	Event driven	Static and Dynamic memory	4.67%

VI. 결론 및 향후 계획

본 연구에서, 우리는 ATRA를 탐지하기 위한 Snoop기반의 커널 검사 시스템을 제안하였다. 이 시스템은 일시적인 공격에 취약한 기존 연구의 단점을 보완하기 위해 Snooper를 이용하여 커널 메모리를 실시간으로 보호하고, 프로세스의 컨텍스트 스위칭 시마다 커널 코드에 대한 페이지를 검사하여 가상-물리 메모리 주소 변환을 조작하는 공격 탐지가 가능하도록 설계하였다. 실험을 통해, 제시된 threat model의 공격탐지가 가능하며, 다른 연구들에 비해 SKIS의 성능저하가 크지 않음을 증명하였다. 향후에는 이 시스템을 이용하여 서비스 제공자가 안전한 환경에서 서비스를 수행할 수 있도록 단말의 변조여부를 전달해주고 단말의 보안정책을 변경하는 Remote attestation 시스템을 추가로 개발할 예정이다.

References

[1] "ARM TrustZone Software Architecture," http://infocenter.arm. com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/EABGFFIC.html

[2] A.M. Azab, P. Ning, J. Shah, Q. Chen, R. Bhutkar, G. Ganesh, J. Ma, and W. Shen, "Hypervision across worlds: real-time kernel protection from the ARM TrustZone secure world," Proceedings of the 21st ACM Conference on Computer and Communications Security, pp. 90-102, Nov. 2014.

〔3〕 X. Ge, H. Vijayakumar, and T. Jaeger, "Sprobes: enforcing kernel code integrity on the TrustZone architecture," Proceedings of Mobile Security Technologies (MoST) 2014, May 2014.
〔4〕 "Sensepost," https://www.sensepost. com/blog/2013/a-software-level-analysis-of-trustzone-os-and-trustlets-in-samsung-galaxy-phone/
〔5〕 Jinmok Kim, Donguk Kim, Jinbum Park, Jihoon Kim, and Hyoungshick Kim, "An Efficient Kernel Introspection System using a Secure Timer on TrustZone," Journal of The Korea Institute of Information Security & Cryptology, 25(4), pp. 863-872, Aug. 2015.
〔6〕 Daehee Jang, Hojoon Lee, Minsu Kim, Daehyeok Kim, Daegyeong Kim and Brent Byunghoon Kang, "ATRA: Address translation redirection attack against hardware-based external monitors," Proceedings of the 21st ACM Conference on Computer and Communications Security, pp. 167-178, Nov. 2014.
〔7〕 R, Niemann, Hardware/Software Co-Design for Data Flow Dominated Embedded Systems, 1998th Ed., Kluwer Academic Publishers, Norwell, MA, USA, 1998.
〔8〕 Hyungon Moon, Hojoon Lee, Jihoon Lee, Kihwan Kim, Yunheung Paek, and Brent Byunghoon Kang, "Vigilare: toward snoop-based kernel integrity monitor," Proceedings of the 19th ACM Conference on Computer and Communications Security, pp. 28-37, Oct. 2012.
〔9〕 R. Riley, X. Jiang, D. Xu, "Multi-aspect profiling of kernel rootkit behavior," Proceedings of the 4th ACM European Conference on Computer Systems, pp. 47-60, Apr. 2009.
〔10〕 Daehee Jang, Jinsoo Jang, Donguk Kim, Changho Choi, and Brent Byunghoon Kang, "Scheduler-based Defense Method against Address Translation Redirection Attack (ATRA)," Journal of The Korea Institute of Information Security & Cryptology, 25(4), pp. 873-880, Aug. 2015.
〔11〕 "AMBA Specification", https://www.arm.com/products/system-ip/amba-specifications.php
〔12〕 "Streamline," https://developer.arm.com/products/software-development-tools/ds-5-development-studio/streamline
〔13〕 "CoreMark," http://www.eembc.org/coremark/about.php
〔14〕 "Unix bench," https://code.google.com/archive/p/byte-unixbench/
〔15〕 "Linpack," http://www.netlib.org/benchmark/hpl/
〔16〕 "nbench," http://www.tux.org/~mayer/linux/bmark.html

〈저 자 소 개〉

김 동 욱 (Donguk Kim) 정회원
2007년 8월: 고려대학교 산업시스템정보공학과 졸업
2007년 7월~2008년 8월: 한국생산성본부 연구원 재직
2014년 2월: KAIST 산업및시스템공학과 박사 졸업
2014년 3월~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 정보보호, 시스템 보안

김 지 훈 (Jihoon Kim) 정회원
2014년 2월: 광운대학교 컴퓨터소프트웨어학과 졸업
2014년 3월~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 정보보호, 시스템 소프트웨어

박 진 범 (Jinbum Park) 정회원
2013년 2월: 경원대학교 컴퓨터소프트웨어학과 졸업
2013년 3월~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 컴퓨터공학, 소프트웨어 보안, 리눅스 커널 보안

김 진 목 (Jinmok Kim) 정회원
2006년 2월: 숭실대학교 정보통신공학부 졸업
2015년 8월: 성균관대학교 정보통신대학 석사 졸업
2005년 12월~현재: 삼성전자 소프트웨어센터 재직
〈관심분야〉 정보보호